

#2

Docket No. 1538.1009/JDH

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)
Yutaka YAMANAKA, et al.)
Serial No.: TO BE ASSIGNED) Group Art Unit: Unassigned
Filed: January 31, 2001) Examiner: Unassigned
For: COMPILER FRO PARALLEL)
COMPUTER)



**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN
APPLICATION IN ACCORDANCE
WITH THE REQUIREMENTS OF 37 C.F.R. §1.55**

*Assistant Commissioner for Patents
Washington, D.C. 20231*

Sir:

In accordance with the provisions of 37 C.F.R. §1.55, the applicant submits herewith a certified copy of the following foreign application:

Japanese Patent Application No. 2000-305608, filed October 5, 2000.

It is respectfully requested that the applicant be given the benefit of the foreign filing date as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. §119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: January 31, 2001

By: _____

James D. Halsey, Jr.
Registration No. 22,729

700 11th Street, N.W., Ste. 500
Washington, D.C. 20001
(202) 434-1500

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application:

2 0 0 0 年 1 0 月 5 日

出 願 番 号
Application Number:

特 願 2 0 0 0 - 3 0 5 6 0 8

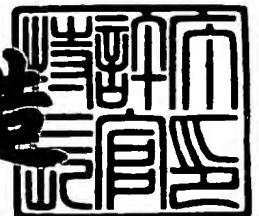
出 願 人
Applicant (s):

富士通株式会社

2 0 0 0 年 1 2 月 1 5 日

特 許 庁 長 官
Commissioner,
Patent Office

及 川 耕 造



出 証 番 号 出 証 特 2 0 0 0 - 3 1 0 4 7 2 0

【書類名】 特許願

【整理番号】 0051274

【提出日】 平成12年10月 5日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 09/45

【発明の名称】 コンパイラ

【請求項の数】 5

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 山中 豊

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 石川 貴洋

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100103528

【弁理士】

【氏名又は名称】 原田 一男

【電話番号】 045-290-2761

【手数料の表示】

【予納台帳番号】 076762

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1
【包括委任状番号】 9909129
【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 コンパイラ

【特許請求の範囲】

【請求項 1】

ソース・プログラムをコンパイルするコンパイラを格納した記録媒体であって

前記コンパイラは、コンピュータに、

前記ソース・プログラム中の並列化指示文を検出するステップと、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップと、

を実行させるためのコンパイラである、記録媒体。

【請求項 2】

前記並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から前記並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップ、

をさらに実行させるためのコンパイラであることを特徴とする請求項 1 記載の記録媒体。

【請求項 3】

各前記処理コードに対応する 1 又は複数の処理情報を格納する処理テーブルを用いて、前記並列化指示文に対するフロントエンド内中間言内の前記処理コードから現在の処理内容に対応する前記処理情報を取得するステップと、

を実行させるためのコンパイラであることを特徴とする請求項 1 記載の記録媒体。

【請求項 4】

前記現在の処理内容が、型解析処理、構文解析処理、意味解析処理、又はコンパイラ内中間言生成であることを特徴とする請求項 3 記載の記録媒体。

【請求項 5】

ソース・プログラムをコンパイルするコンパイル装置であって、
前記ソース・プログラム中の並列化指示文を検出する手段と、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成する手段と、

を有するコンパイル装置。

【発明の詳細な説明】

【 0 0 0 1 】

【発明が属する技術分野】

本発明は、コンパイラに関し、より詳しくは並列コンピュータに対するソース・プログラムをコンパイルするコンパイラに関する。

【 0 0 0 2 】

【従来の技術】

近年、複数のCPU (Central Processing Unit) を有するコンピュータは、CPU価格の下落等の理由で普及してきている。よって、例えばOpenMPといった、共有メモリ型の並列コンピュータ向けプログラミングにおけるAPI (Application Program Interface) の使用も広がってきている。このような状況の下、並列コンピュータのためのコンパイラの重要性も上がってきている。

【 0 0 0 3 】

【発明が解決しようとする課題】

しかし、従来では並列化指示文がソース・プログラムにおいて指定されていた時に、コンパイラでは並列化指示文内の構造情報を保持することなく処理していた。また、並列化指示文と当該並列化指示文が有効となる文 (式を含む) との関係を示す情報を保持することなく処理していた。このため、コンパイラにおける処理が複雑になり、且つ速度の向上を阻害していた。

【 0 0 0 4 】

よって本発明の目的は、新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供することである。

【 0 0 0 5 】

【課題を解決するための手段】

本発明に係る、並列化指示文を含むソース・プログラムをコンパイルするコンパイラは、コンピュータに、当該ソース・プログラム中の並列化指示文を検出するステップと、

【 0 0 0 6 】

並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップとを実行させる。

【 0 0 0 7 】

これにより構文解析、意味解析などの解析処理で階層構造を有するフロントエンド内中間言を使用することができるようになり、その都度階層構造を確認しながら解析処理を行っていた従来よりも処理が高速化される。

【 0 0 0 8 】

また、並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から当該並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップをさらに実行させるようにする。これにより、並列化指示文とそれが有効となる文との関係が明確となる。使用する記憶領域を削減する効果もある。

【 0 0 0 9 】

なお、上で述べたようなコンパイラが通常のコンピュータで実行されれば当該コンピュータはコンパイル装置となる。また、コンパイラは、例えばフロッピーディスク、CD-ROM、光磁気ディスク、半導体メモリ、ハードディスク等の記憶媒体又は記憶装置に格納される。また、コンパイラの処理途中の中間的なデータは、コンピュータのメインメモリなどの記憶装置に格納される。

【 0 0 1 0 】

【発明の実施の形態】

本発明の一実施の形態に係る機能ブロック図を図 1 に示す。コンピュータ 1 は

、OpenMPなどのAPIに従った並列化指示文を含むソースプログラムのファイル10をコンパイルし、生成したオブジェクトコードのファイル30を出力するコンパイラ20を実行する。コンパイラ20には、プログラミング言語に従って書かれたソースプログラムに含まれる各字句を解析する字句解析部21と、ソースプログラムの構造を解析する構文解析部22と、ソースプログラムの意味内容を解析する意味解析部23と、中間言変換部24とが含まれる。この字句解析部21と構文解析部22と意味解析部23と中間言変換部24はフロントエンド部40と呼ばれ、フロントエンド部40内で用いられる中間言をフロントエンド内中間言と呼ぶ。中間言変換部24は、フロントエンド内中間言を最適化処理などで用いるコンパイラ内中間言（単に「中間言」と呼ばれることもある。）に変換する。またコンパイラ20には、最適化処理を行う最適化処理部25と、最適化処理などの結果を用いてオブジェクトコードを生成するコード生成部26とがさらに含まれる。

【0011】

本実施の形態では構文解析部22に以下で説明する処理を追加することにより、OpenMPなどに従って記述された並列化指示文をリスト構造化して、フロントエンド内中間言を生成する。リスト構造は階層構造の一種である。

【0012】

例えば、

```
#pragma omp parallel private(a,b) lastprivate(c)
```

上の並列化指示文が有効となる文

といった文がソースプログラムに含まれている場合には、図2に示すようなデータ構造を生成する。なお、「#pragma omp」はOpenMPのAPIであることを示すものである。また、上の例ではparallelが指示子（directive）、private及びlastprivateが節（clause）、a、b及びcが並びと呼ばれている。

【0013】

図2では、並列化指示文が有効となる文のコード格納域201と、その文についての各種情報格納域203とが含まれる。例えば、並列化指示文がFOR文で

あれば、コード格納域 2 0 1 には F O R 文であることを示すコードが格納され、各種情報格納域 2 0 3 にはループの情報等が格納される。なお、ここまでは従来と変わらない。この領域を `a_statement` と呼ぶ。本実施の形態では、並列化指示文の指示子（上の例では `parallel`）のアドレス格納域 2 0 5 が、並列化指示文が有効となる文のフロントエンド内中間言において付加されている。これにより、以下に示すリスト構造へのアクセスが容易になり、並列化指示文と当該並列化指示文が有効となる文との関係が明確化される。また、並列化指示文の存在のみを示すフロントエンド内中間言が不要になる。

【 0 0 1 4 】

指示子のアドレス格納域 2 0 5 に格納されているアドレス（* 1）が、指示子 `parallel` に関連する節へのリンク情報等が格納される情報域 2 0 7 の先頭となる。また、情報域 2 0 7 に対応して、`parallel` に対応する処理コードが格納される情報域 2 0 9 も設けられている。また、節 `private` に関連する並びへのリンク情報及び節 `lastprivate` へのリンク情報等が格納される情報域 2 1 1 と、`private` に対応する処理コードが格納される情報域 2 1 3 とが対応して設けられている。さらに、節 `lastprivate` に関連する並びへのリンク情報等が格納される情報域 2 1 5 と、`lastprivate` に対応する処理コードが格納される情報域 2 1 7 とが対応して設けられる。

【 0 0 1 5 】

節 `private` に関連する並び `a` に関連する他の並びへのリンク情報等が格納される情報域 2 1 9 と、`a` に対応する処理コードが格納される情報域 2 2 1 とが対応して設けられる。また、節 `private` に関連する並び `b` に関する情報が格納される情報域 2 2 3（上の例では空）と、`b` に対応する処理コードが格納される情報域 2 2 5 とが対応して設けられる。また、節 `lastprivate` に関連する並び `c` に関する情報が格納される情報域 2 2 7（上の例では空）と、`c` に対応する処理コードが格納される情報域 2 2 9 とが対応して設けられる。なお、処理コードは、指示子、節、並びの種類を区別することも可能な一意のコード（数値）である。例えば `parallel` は 0 x 0 1、`private` は 0 x 0 2 である。

【 0 0 1 6 】

このようなデータ構造を生成するために、従来の処理フローに加えて実施される処理フローを図3を用いて説明する。最初に並列化指示文か否かを判断する（ステップS1）。並列化指示文でなければ元の処理に戻る。並列化指示文である場合には、指示子があるか判断する（ステップS3）。並列化指示文であって指示子がない場合には、エラーであるから元の処理に戻る。もし指示子がある場合には、指示子の処理コードをリスト構造に設定する（ステップS5）。これがリスト構造の先頭領域となる。なお図2の例ではparallelの処理コードを設定する。そして、並列化指示文が有効となる文のフロントエンド中間言に、リスト構造の起点を登録する（ステップS7）。図2の例では指示子のアドレス格納域205に、指示子parallelの情報域207のアドレスを格納する。

【0017】

以下の説明では、情報域207及び209、情報域211及び213、情報域215及び217、情報域219及び221、情報域223及び225、情報域227及び229をそれぞれ1つの情報域として説明する場合もある。

【0018】

次に未処理の節が存在するか判断する（ステップS9）。未処理の節が存在しない場合には、元の処理に戻る。もし未処理の節が存在している場合には、節の処理コードをリスト構造に設定する（ステップS11）。なお、節の処理コードをリスト構造に設定した場合には、最初は当該節に関連する指示子の情報域に当該節の情報域のアドレスを格納する。また、節が2つ以上ある場合において、2番目以降の節については前の節の情報域に当該節の情報域のアドレスを格納する。図2の例では、指示子parallelに関連する節はprivateとlastprivateの2つあり、節privateの情報域211のアドレスは指示子parallelの情報域207に格納され、節lastprivateの情報域215のアドレスは節privateの情報域211に格納される。

【0019】

そして未処理の並びが存在するか判断する（ステップS13）。もし未処理の並びがなければステップS9に戻る。一方、未処理の並びが存在している場合には、並びの処理コードをリスト構造に設定する（ステップS15）。なお、並び

の処理コードをリスト構造に設定した場合には、最初は当該並びに関連する節の情報域に当該並びの情報域のアドレスを格納する。また、並びが2つ以上ある場合において、2番目以降の並びについては前の並びの情報域に当該並びの情報域のアドレスを格納する。図2の例では、節privateに関連する並びはaとbの2つあり、並びaの情報域219のアドレスは節privateの情報域211に格納され、並びbの情報域223のアドレスは並びaの情報域219に格納される。なお、節lastprivateに関連する並びはcだけであり、並びcの情報域227のアドレスは節lastprivateの情報域215に格納される。

【0020】

ステップS13及びステップS15は未処理の並びが無くなるまで繰り返される。図3に示すような処理により図2に示されるようなデータ構造が生成される。

【0021】

図2に示されるようなデータ構造は、図4に示すような処理テーブルと合わせて使用される。図4の処理テーブルでは、列401が処理1についての情報を格納しており、列403が処理2についての情報を格納している。また、行405がparallelの処理コードに対応する、処理1、処理2...についての情報を格納しており、行407がprivateの処理コードに対応する、処理1、処理2...についての情報を格納しており、行409がvariableの処理コードに対応する処理1、処理2...についての情報を格納している。

【0022】

処理1、処理2といった処理は、例えば構文解析の情報取得又は処理関数実行や、意味解析の情報取得又は処理関数実行、型チェックの情報取得又は処理関数実行、コンパイラ内中間言の取得といった、構文解析部22、意味解析部23及び中間言変換部24において実行される各種処理である。

【0023】

例えば、処理1を型チェックのための情報取得とし、例えば図4のa1をon、図4のa2をoffとする。そうすると、型チェックのための情報取得を行う際には、parallelの処理コードに対してはonという情報が、privateの処理コ

ードに対しては `off` という情報が取得できる。同様に、処理 2 を意味解析用関数アドレスとし、図 4 の `b 1` を `parallel` 用の意味解析用関数アドレスとし、`b 2` を `private` 用の意味解析用関数アドレスとする。そうすると意味解析処理を行う際には、`parallel` の処理コードに対して `parallel` 用の意味解析用関数アドレスを取得することができ、その関数を実行することができる。また、`private` の処理コードに対しては `private` 用の意味解析用関数アドレスを取得することができ、その関数を実行することができる。

【 0 0 2 4 】

本実施の形態では、処理テーブル内の情報を取得する場合には、例えば以下のような式を用いる。

テーブル情報 = 処理テーブル [処理コード] . 処理 x

ここで処理 x とは、処理 1、処理 2 . . . を表す。このように処理テーブルから取得した「テーブル情報」を用いて処理を行うことができる。例えば、処理 1 がコンパイラ内中間言の取得であって、`a 1` が `parallel` (処理コード `0x01`) のコンパイラ内中間言である `parallel_x` であり、`a 2` が `private` (処理コード `0x02`) のコンパイラ内中間言である `private_x` とすると、

`parallel` コンパイラ内中間言 = 処理テーブル [`0 x 0 1`] . 処理 1

で `parallel` コンパイラ内中間言に `parallel_x` が入る。

`private` コンパイラ内中間言 = 処理テーブル [`0 x 0 2`] . 処理 1

で `private` コンパイラ内中間言に `private_x` が入る。

【 0 0 2 5 】

一例として中間言変換部 2 4 における、処理テーブル及び処理コードを用いたフロントエンド内中間言からコンパイラ内中間言への変換処理（本実施の形態における変更分）を図 5 を用いて説明する。

【 0 0 2 6 】

最初にリスト構造を有する文のフロントエンド内中間言であるか否かを判断する（ステップ `S 2 1`）。本実施の形態の場合、指示子のアドレス格納域が存在しているか検査すれば良い。もし、リスト構造を有する文のフロントエンド内中間言でなければ、元の処理に戻る。リスト構造を有する文のフロントエンド内中間

言であれば、並列化指示文のコンパイラ内中間言を生成する（ステップ S 2 3）。並列化指示文のコンパイラ内中間言は、従来と同じである。

【 0 0 2 7 】

そして指示子が存在しているか判断する（ステップ S 2 5）。指示子が存在しているかは、リスト構造において最初の処理コードを調べればよい。処理コードは、指示子、節、並びの別をも示している。もし、指示子が存在していなければ元の処理に戻る。指示子が存在していれば、指示子のコンパイラ内中間言を処理テーブルを用いて設定する（ステップ S 2 7）。ここでは、指示子コンパイラ内中間言＝処理テーブル [指示子の処理コード] . 処理 1（処理 1 がコンパイラ内中間言の取得である場合）という処理を行う。

【 0 0 2 8 】

次に未処理の節が存在しているか判断する（ステップ S 2 9）。ステップ S 2 9 の最初の実施の場合には、リスト構造内の指示子の情報域に格納されている他の情報域のアドレスが存在しているか否かで判断する。ステップ S 2 9 の実施が 2 度目以降の場合には、前の節の情報域内に、並びの情報域以外の情報域のアドレスが格納されているかで判断する。もし、未処理の節が存在していない場合には、元の処理に戻る。未処理の節が存在している場合には、節のコンパイラ内中間言を処理テーブルを用いて設定する（ステップ S 3 1）。節の情報域のアドレスにアクセスし、その節の処理コードを取得する。そして、節コンパイラ内中間言＝処理テーブル [節の処理コード] . 処理 1 という処理を行う。

【 0 0 2 9 】

次に、未処理の並びが存在しているか判断する（ステップ S 3 3）。ステップ S 3 3 の最初の実施の場合には、リスト構造内の節の情報域内に、他の情報域のアドレスが存在するか確認し、もし存在する場合には当該他の情報域のアドレスにアクセスしてその情報域に格納された処理コードが並びの処理コードか否かで判断する。また、ステップ S 3 3 の 2 度目以降の実施の場合には、現在の並びの情報域に他の情報域へのアドレスが格納されているかで判断する。もし、未処理の並びが存在していない場合には、ステップ S 2 9 に戻る。一方、未処理の並びが存在している場合には、並びのコンパイラ内中間言を処理テーブルを用いて設

定する（ステップ S 3 5）。並びの情報域のアドレスにアクセスし、並びの処理コードを取得する。そして、並びコンパイラ内中間言＝処理テーブル [並びの処理コード]．処理 1 という処理を行う。

【 0 0 3 0 】

以上のような処理を行うことにより、フロントエンド内中間言からコンパイラ内中間言を生成することができるようになる。指示子、節及び並びの処理コードを含むリスト構造と処理テーブルを保持しているため、無駄な解析処理を行わずに済むので、簡単且つ高速にコンパイラ内中間言を処理テーブルから取得することができる。

【 0 0 3 1 】

なお、ステップ S 2 5 以降の処理は、他の処理、例えば、意味解析用情報の取得や、意味解析処理の実施（意味解析用関数関数アドレスの取得と実行）等の場合でも、ステップ S 2 7、ステップ S 3 1 及びステップ S 3 3 を「テーブル情報＝処理テーブル [処理コード]．処理 x」といった形の処理に変更すれば、他の処理に適用することができる。その場合には、無駄な解析処理を行わずに済むので、簡単且つ高速に所望の処理を実施することができるようになる。また、新たな処理が必要になった場合には、処理テーブルの列の数を増やせば簡単に対処することができるようになる。

【 0 0 3 2 】

以上本発明の一実施の形態を説明したが、様々な変形が可能である。例えば図 1 において最適化処理部 2 5 の後にコード生成部 2 6 が設けられているが、最適化処理の他に他の処理（例えばレジスタ割付処理）を実施した後にコード生成処理を行うような構成であってもよい。また、リスト構造は図 2 のような構造でなくとも、並列化指示文の構造が表された階層的なデータ構造であっても良い。また、図 3 及び図 5 の処理フローは O p e n M P における並列化指示文を前提とした処理フローとなっているので、他のルールに従った並列化指示文の場合にはその並列化指示文の構造に従って変更する必要がある。上でも述べたが図 4 の処理テーブルは、列を追加すれば様々な処理に対処できるようになり、行を増やせば取扱い処理コードの数が増える。

【 0 0 3 3 】

また、図 1 のコンピュータ 1 がネットワークに接続されており、ソースプログラム・ファイル 1 0 が他のコンピュータから送信されてきたり、オブジェクトコード・ファイル 3 0 が他のコンピュータへ送信されるような場合もある。他のコンピュータは並列コンピュータである場合もある。

【 0 0 3 4 】

さらに上で述べたコンパイラは通常のコンピュータにおいて実行されるプログラムの形態で実施され、その場合、コンパイラは、例えばフロッピーディスク、CD-ROM、光磁気ディスク、半導体メモリ、ハードディスク等の記憶媒体又は記憶装置に格納される。プログラムであるコンパイラは、ネットワークを介して配布される場合もある。

【 0 0 3 5 】

(付記 1)

ソース・プログラムをコンパイルするコンパイラを格納した記録媒体であって

前記コンパイラは、コンピュータに、

前記ソース・プログラム中の並列化指示文を検出するステップと、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップと、

を実行させるためのコンパイラである、記録媒体。

【 0 0 3 6 】

(付記 2)

前記並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から前記並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップ、

をさらに実行させるためのコンパイラであることを特徴とする付記 1 記載の記録媒体。

【 0 0 3 7 】

(付記 3)

各前記処理コードに対応する 1 又は複数の処理情報を格納する処理テーブルを用いて、前記並列化指示文に対するフロントエンド内中間言内の前記処理コードから現在の処理内容に対応する前記処理情報を取得するステップと、

を実行させるためのコンパイラであることを特徴とする付記 1 記載の記録媒体

。

【 0 0 3 8 】

(付記 4)

前記現在の処理内容が、型解析、構文解析、意味解析、又はコンパイラ内中間言生成であることを特徴とする付記 3 記載の記録媒体。

【 0 0 3 9 】

(付記 5)

前記階層構造が、リスト構造であることを特徴とする付記 1 記載の記録媒体。

【 0 0 4 0 】

(付記 6)

前記並列化指示文の部分には、指示子、節及び並びが含まれ、

前記指示子の処理コードの下位に前記節の処理コード、前記節の処理コードの下位に並びの処理コードがリンク付けされる

ことを特徴とする付記 1 記載の記録媒体。

【 0 0 4 1 】

(付記 7)

ソース・プログラムをコンパイルするコンパイル装置であって、

前記ソース・プログラム中の並列化指示文を検出する手段と、

前記並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成する手段と、

を有するコンパイル装置。

【 0 0 4 2 】

【発明の効果】

以上のように、新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供することができた。

【図面の簡単な説明】

【図 1】

本発明の一実施の形態に係るコンパイラを実行するコンピュータの機能ブロック図である。

【図 2】

本発明の一実施の形態に係るフロントエンド内中間言の一例を示す図である。

【図 3】

図 2 のデータ構造を生成するための処理フローを示す図である。

【図 4】

処理テーブルの一例を示す図である。

【図 5】

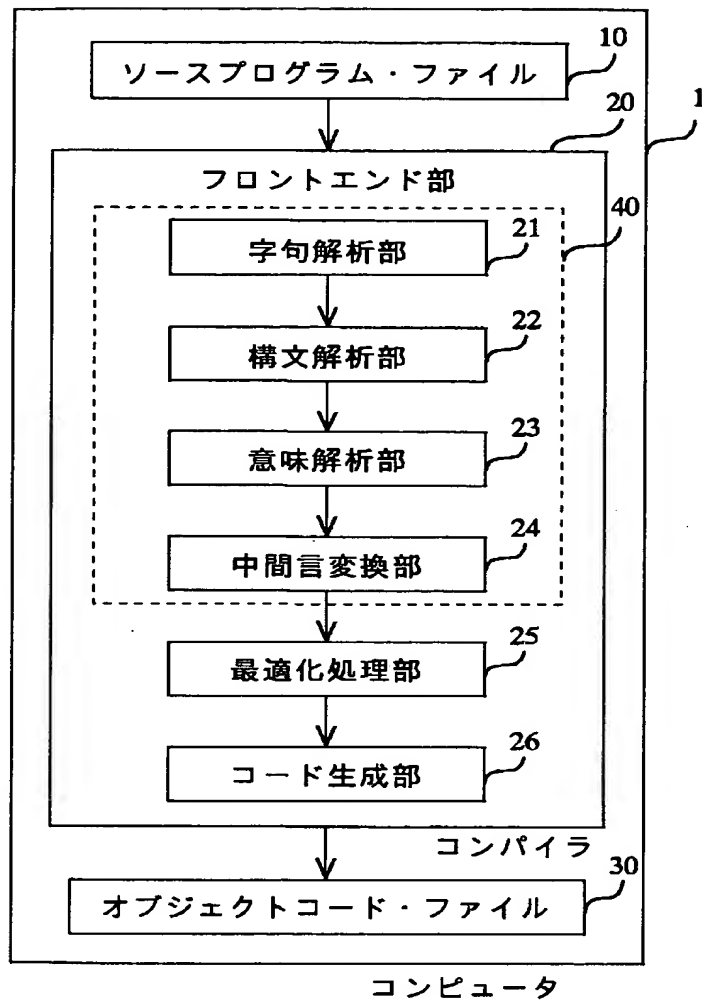
フロントエンド内中間言からコンパイラ内中間言を生成するための処理フローを表す図である。

【符号の説明】

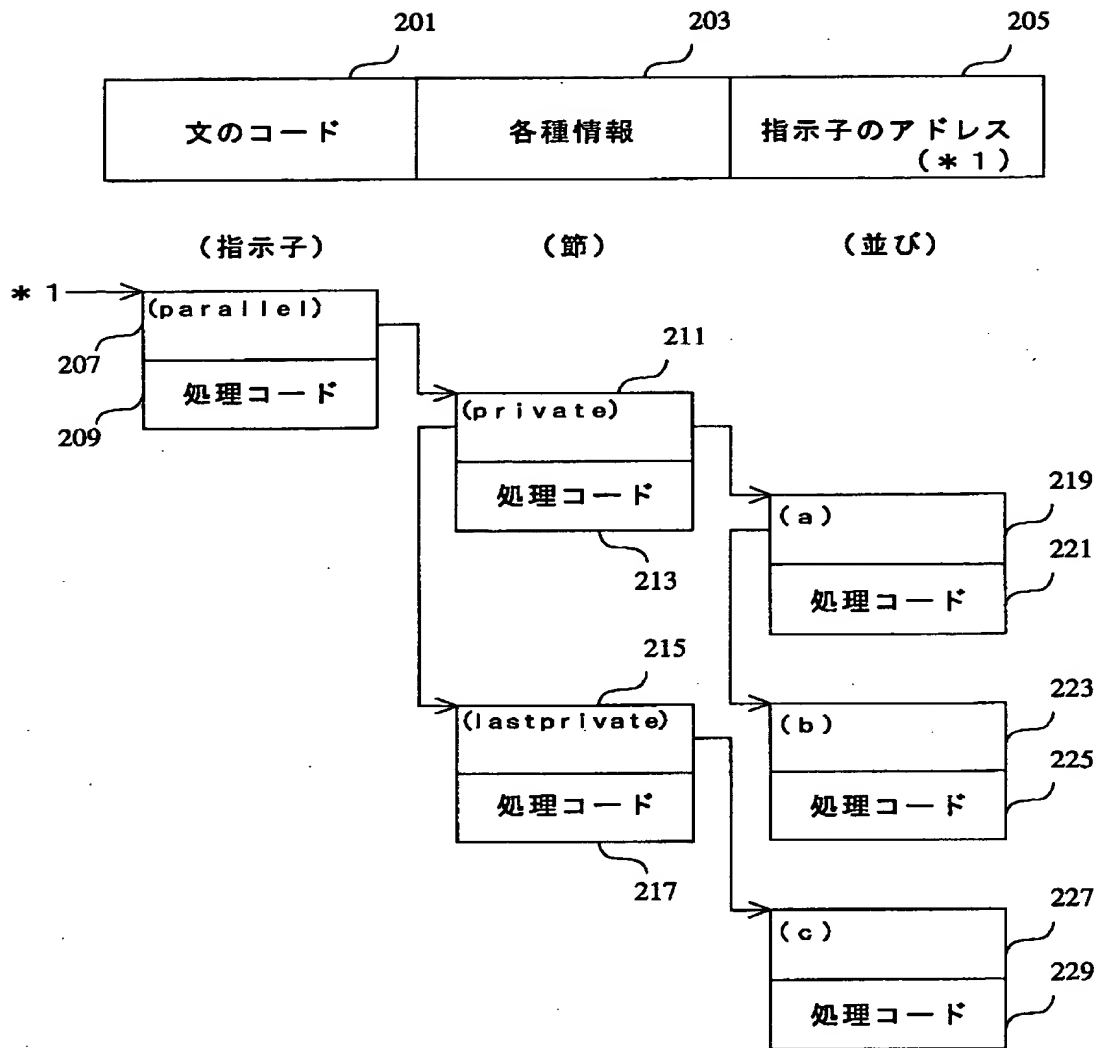
- | | | | | | |
|-----|----------|-----|----------------|-----|--------|
| 1 | コンピュータ | 1 0 | ソースプログラム・ファイル | | |
| 2 0 | コンパイラ | 2 1 | 字句解析部 | 2 2 | 構文解析部 |
| 2 3 | 意味解析部 | 2 4 | 中間言変換部 | 2 5 | 最適化処理部 |
| 2 6 | コード生成部 | 3 0 | オブジェクトコード・ファイル | | |
| 4 0 | フロントエンド部 | | | | |

【書類名】 図面

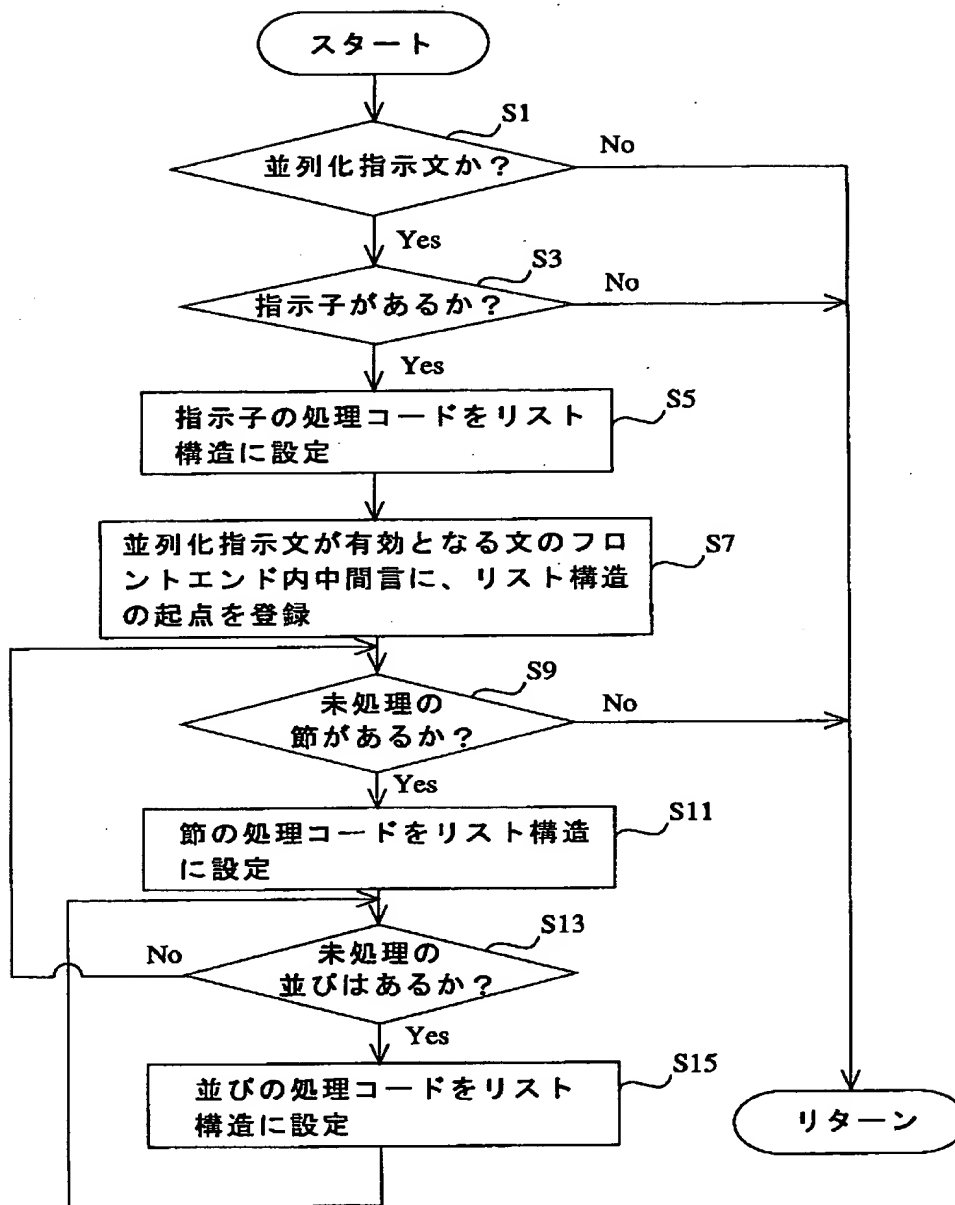
【図 1】



【図 2】



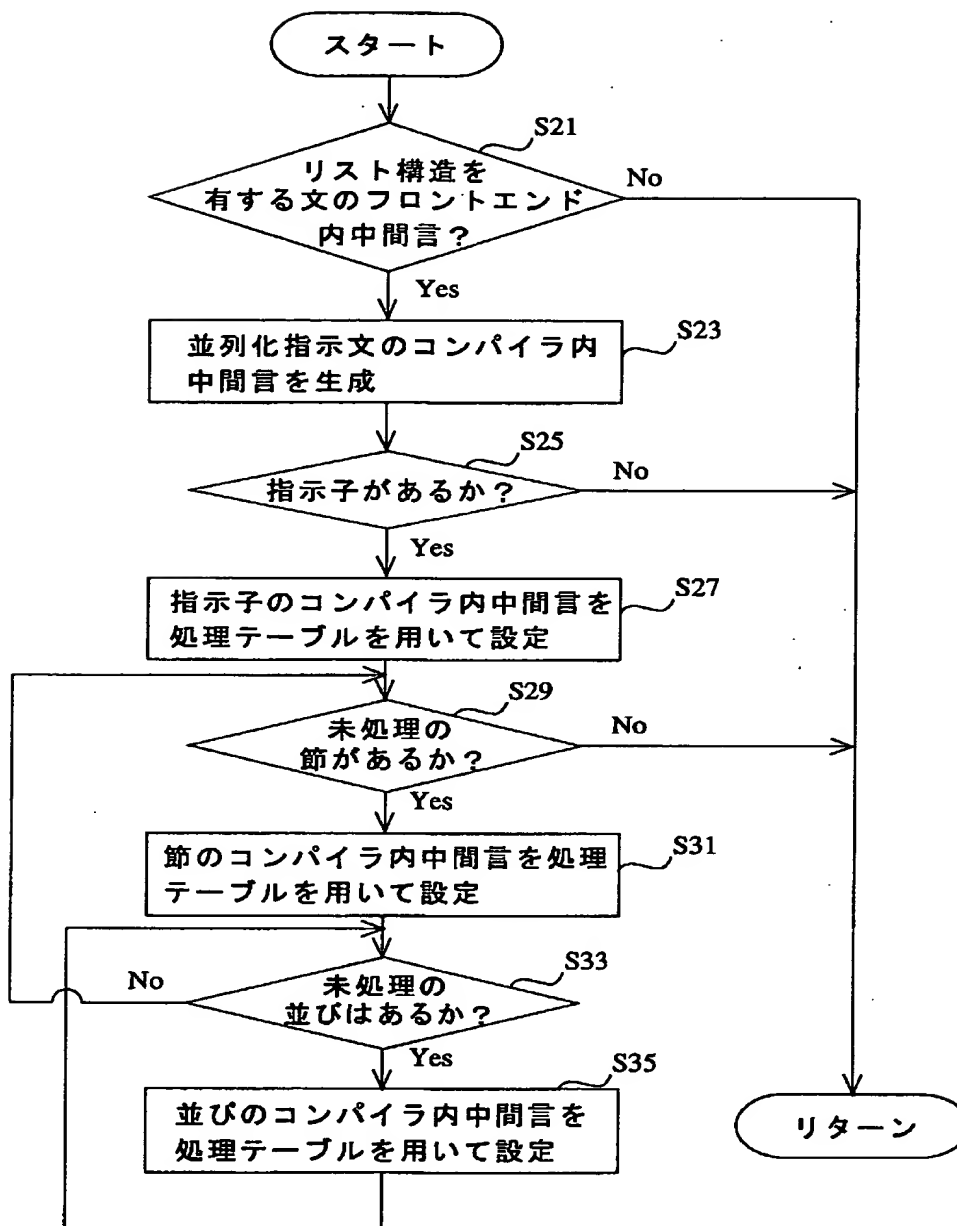
【図 3】



【図 4】

<div> <div>401</div> <div>403</div> </div>		処理	
		処理 1	処理 2
処理コード			
405	parallel	a1	b1
407	private	a2	b2
409	variable	a3	b3
411	lastprivate	a4	b4
	⋮	⋮	⋮

【図 5】



【書類名】 要約書

【要約】

【課題】

新規のデータ構造を導入して高速処理可能な、並列コンピュータのためのコンパイラを提供する。

【解決手段】

並列化指示文を含むソース・プログラム10をコンパイルするコンパイラ20は、ソース・プログラム中の並列化指示文を検出するステップと、並列化指示文を検出した場合に、当該並列化指示文内部の構造に従って当該並列化指示文の少なくとも一部分の各処理コードを階層構造をもって記憶領域上に配置することにより、並列化指示文に対するフロントエンド内中間言を生成するステップとを実行する。また、並列化指示文が有効となる文のフロントエンド内中間言に、当該並列化指示文が有効となる文のフロントエンド内中間言から当該並列化指示文に対するフロントエンド内中間言への参照情報を付加するステップも実行する。

【選択図】 図3

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 2 2 3]

1. 変更年月日	1 9 9 6 年 3 月 2 6 日
[変更理由]	住所変更
住 所	神奈川県川崎市中原区上小田中 4 丁目 1 番 1 号
氏 名	富士通株式会社